

Detecting Short Reset Words Parameterized by Number of States^{*}

Vojtěch Vorel

Charles University in Prague, Czech Republic
vorel@ktiml.mff.cuni.cz

Abstract. A word is called a reset word for a deterministic finite automaton if it maps all states of the automaton to a unique state. Deciding about the existence of a reset word of a given length for a given automaton is known to be a NP-complete problem. We use tools provided by Bodlaender et al. (2007) to prove that such problem, parameterized by number of states, does not admit a polynomial kernel unless polynomial hierarchy collapses. This completes the multivariate analysis of the problem started by Fernau et al. (2013).

1 Introduction

1.1 Reset Words and Synchronization

A *deterministic finite automaton* is a triple $A = (Q, X, \delta)$, where Q and X are finite sets and δ is an arbitrary mapping $Q \times X \rightarrow Q$. Elements of Q are called *states*, X is the *alphabet*. The *transition function* δ can be naturally extended to $Q \times X^* \rightarrow Q$, still denoted by δ . We extend it also by defining

$$\delta(S, w) = \{\delta(s, w) \mid s \in S, w \in X^*\}$$

for each $S \subseteq Q$. If an automaton $A = (Q, X, \delta)$ is fixed, we write

$$r \xrightarrow{x} s$$

instead of $\delta(r, x) = s$.

For a given automaton $A = (Q, X, \delta)$, we call $w \in X^*$ a *reset word* if

$$|\delta(Q, w)| = 1.$$

If such a word exists, we call the automaton *synchronizing*. Note that each word having a reset word as a factor is also a reset word.

The *Černý conjecture*, a longstanding open problem, claims that each synchronizing automaton has a reset word of length at most $(|Q| - 1)^2$. There is a series of automata due to Černý whose shortest reset words reach this bound exactly [2], but all the known upper bounds lie in $\Omega(|Q|^3)$. A tight bound has been established for various special classes of automata, see some of recent advances in [5,8].

^{*} Research supported by the Czech Science Foundation grant GA14-10799S.

Namely, the best general upper bound of the length of shortest reset words is currently the following¹:

Lemma 1 ([6]). *Any n -state synchronizing automaton has a reset word of length $z(n)$, where*

$$z(n) = \left\lfloor \frac{n^3 - n}{6} \right\rfloor.$$

It is convenient to see synchronization as a process in discrete time. Having an automaton $A = (Q, X, \delta)$ and a word

$$w = x_1 \dots x_{|w|}$$

fixed, we say that a state $s \in S$ is *active in time* $l \leq |w|$ if

$$s \in \delta(Q, x_1 \dots x_l).$$

In time 0, before the synchronization starts, all the states are active. As we apply the letters, the number of active states may decrease. We may consider that active states are identified by *activity markers*, which move along appropriate transitions whenever a letter is applied. If two activity markers meet each other, they just merge.

When the number of active states decreases to 1 at a time l , synchronization is complete and the word $x_1 \dots x_l$ is a reset word.

1.2 Computational tasks

Let us define two of fundamental decision problems related to reset words. The first is just to decide, if a given automaton is synchronizing:

EX-SYN

Input: synchronized automaton $A = ([n], X, \delta)$

Output: does A have a reset word?

The second problem is closely related to EX-SYN, but now there is a need to find a *short* reset word:

SYN

Input: synchronized automaton $A = ([n], X, \delta)$, $d \in \mathbb{N}$

Output: does A have a reset word of length d ?

Lemma 2 ([2]). *There is a polynomial-time algorithm for EX-SYN.*

Corollary 3. *SYN, if restricted to the instances with $d \geq z(n)$, is solvable in polynomial time.*

Lemma 4 ([3]). *SYN is NP-complete. Even if restricted to automata with two-letter alphabets.*

¹ An improved bound published by Trakhtman [9] in 2011 has turned out to be proved incorrectly.

1.3 Parameterized Complexity

We adopt formalizations used in [1]. *Parameterized problem* L is any subset of pairs (x, n) from $\Sigma^* \times \mathbb{N}$ for a finite alphabet Σ . Let $L \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. Its *unparameterized version* is

$$\widehat{L} = \{x\#a^n \mid (x, n) \in L\}$$

where $\# \notin \Sigma, a \in \Sigma$.

We consider a variant of SYN that is parameterized by the number of states. It is defined as the following set of tuples $((A, d), n)$ from $\Sigma^* \times \mathbb{N}$, assuming a straightforward encoding of the transition table over an alphabet Σ :

$$L_{\text{SYN}} = \{((A, d), n) \mid A \text{ is } n\text{-state and has a reset word of length } d\}.$$

The following lemma, which is easy to prove using the construction of power automaton, says that SYN is *fixed parameter tractable (FPT)* if parameterized by number of states:

Lemma 5 ([4, 7]). *There is an algorithm for deciding about $((A, d), n) \in L_{\text{SYN}}$, with $A = (Q, X, \delta)$, in time $r(n, |X|) \cdot 2^n$ for an appropriate polynomial r .*

Clearly, the problems SYN, L_{SYN} and \widehat{L}_{SYN} are just different formalizations of the same problem. Adding the unary record of n to the input does not change the computational complexity. However, SYN is suitable for the general introduction and $L_{\text{SYN}}, \widehat{L}_{\text{SYN}}$ are needed to use the results of [1] directly.

Definition 6. *A kernel for L is an algorithm that takes any $(x, n) \in \Sigma^* \times \mathbb{N}$ and outputs in time $p(|x|, n)$ a pair $(x', n') \in \Sigma^* \times \mathbb{N}$ such that:*

- $(x, n) \in L \Leftrightarrow (x', n') \in L$
- $|x'|, n' \leq f(n)$

where p is polynomial and f is computable. Such kernel is polynomial if f is a polynomial.

1.4 Composition Algorithms

Definition 7. *A composition algorithm for a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that*

- receives as input a sequence $((x_1, n), \dots, (x_t, n))$ with $(x_i, n) \in \Sigma^* \times \mathbb{N}^+$ for each $1 \leq i \leq t$,
- uses time polynomial in $\sum_{i=1}^t |x_i| + n$
- outputs $(y, n') \subseteq \Sigma^* \times \mathbb{N}^+$ with
 1. $(y, n') \in L \Leftrightarrow$ there is some $1 \leq i \leq t$ with $(x_i, n) \in L$,
 2. n' is polynomial in n .

Theorem 8 ([1]). *Let L be a parameterized problem having a composition algorithm. Assume that its unparameterized version \widehat{L} is NP-complete. If L has a polynomial kernel, then $\text{PH} = \Sigma_p^3$.*

By PH we denote the union of entire polynomial hierarchy, so $\text{PH} = \Sigma_p^3$ means that polynomial hierarchy collapses into the third level.

2 Main Result

We prove that L_{SYN} does not have a polynomial kernel unless $\text{PH} = \Sigma_p^3$. Due to Theorem 8 it is enough to describe a composition algorithm for L_{SYN} .

2.1 Preprocessing

Receive an input

$$((A_1, d_1), n), \dots, ((A_t, d_t), n)$$

consisting of n -state automata A_1, \dots, A_t , each of them equipped with a number d_i . Assume that the following easy procedures have been already applied:

- For each $i = 1, \dots, t$ such that $d_i \geq z(n)$, use the polynomial-time synchronizability algorithm from Corollary 2 to decide whether $((A_i, d_i), n) \in L_{\text{SYN}}$. If it is, output a trivial true instance immediately. Otherwise just delete the i -th member from the sequence.
- For each $i = 1, \dots, t$, add an additional letter κ to the automaton A_i such that κ acts as the identical mapping: $\delta_i(s, \kappa) = s$.
- For each $i = 1, \dots, t$ rename the states and letters of A_i such that

$$\begin{aligned} A_i &= (Q_i, X_i, \delta_i) \\ Q_i &= \{1, \dots, n\} \\ X_i &= \{\kappa, a_{i,1}, \dots, a_{i,|X_i|-1}\}. \end{aligned}$$

Choose one of the following procedures according to the length t of the input sequence:

- If $t \geq 2^n$, use the exponential-time algorithm from Lemma 5: Denote $D = \sum_{i=1}^t |(A_i, d_i)| + n$, where we add lengths of descriptions of the pairs. Note that $D \geq t \geq 2^n$ and that D is the quantity used to restrict the running time of composition algorithms. By the lemma, in time

$$\sum_{i=1}^t r(n, |X_i|) \cdot 2^n \leq t \cdot r(D, D) \cdot 2^n \leq D^2 \cdot r(D, D)$$

we are able to analyze all the t automata and decide if some of them has a reset word of the corresponding length. It remains just to output some appropriate trivial instance $((A', d'), n')$.

- If $t < 2^n$, we denote $q(t) = \lfloor \log(t+1) \rfloor$. It follows that $q(t) \leq n+2$. On the output of the composition algorithm we put $((A', d'), n')$, where A' is the automaton described in the following paragraphs and

$$d' = z(n) + 1$$

is our choice of the maximal length of reset words to be found for A' .

2.2 Construction of A' and Its Ideas

We set

$$\begin{aligned} A' &= (Q', X', \delta') \\ Q' &= \{1, \dots, n\} \cup \{D\} \cup (\{0, \dots, z(n)\} \times \{0, \dots, q(t)\} \times \{T, F\}) \\ X' &= \left(\bigcup_{i=1}^t X_i \right) \cup \{\alpha_1, \dots, \alpha_t\} \cup \{\omega_1, \dots, \omega_n\}. \end{aligned}$$

On the states $\{1, \dots, n\}$ the letters from $\bigcup_{i=1}^t X_i$ act simply:

$$s \xrightarrow{x_{i,j}} \delta_i(s, x_{i,j})$$

for each $s \in 1, \dots, n$, $i = 1, \dots, t$, $j = 1, \dots, |X_i|$. In other words, we let all the letters from all the automata A_1, \dots, A_t act on the states $1, \dots, n$ just as they did in the original automata. The additional letters act on $\{1, \dots, n\}$ simply as well:

$$s \xrightarrow{\alpha_i} s \quad s \xrightarrow{\omega_{\bar{s}}} \begin{cases} D & \text{if } \bar{s} = s \\ s & \text{otherwise.} \end{cases}$$

for each $s, \bar{s} \in 1, \dots, n$, $i = 1, \dots, t$. The state D is *absorbing*, which means that

$$D \xrightarrow{y} D$$

for any $y \in X'$. Note that any reset word of A' have to map all the states of Q' to D .

The remaining $2 \cdot (z(n) + 1) \cdot (q(t) + 1)$ states form what we call *guard table*. Its purpose is to guarantee that:

- (C1) Any reset word of A' have to be of length at least $d' = z(n) + 1$.
- (C2) Any reset word w of A' , having length exactly $z(n) + 1$, is of the form

$$w = \alpha_i y_1 \dots y_{d_i} \kappa^{z(n)-1-d_i} \omega_s \quad (1)$$

for some $i \in \{1, \dots, t\}$, $y_1, \dots, y_{d_i} \in X_i$, and $s \in \{1, \dots, n\}$, such that $y_1 \dots y_{d_i}$ is a reset word of A_i .

- (C3) Any word w
 - of length $d' = z(n) + 1$,
 - of the form (1),
 - and satisfying that $\delta_i(Q_i, y_1 \dots y_{d_i}) = \{s\}$
is a reset word of A' .

If the guard table manages to guarantee these three properties of A' , we are done: Is is easy to check that they imply all the conditions given in Definition 7. So, let us define the action of the letters from X' on the states from $\{0, \dots, z(n)\} \times \{0, \dots, q(t)\} \times \{T, F\}$. After that the automaton A' will be complete and we will check the three properties.

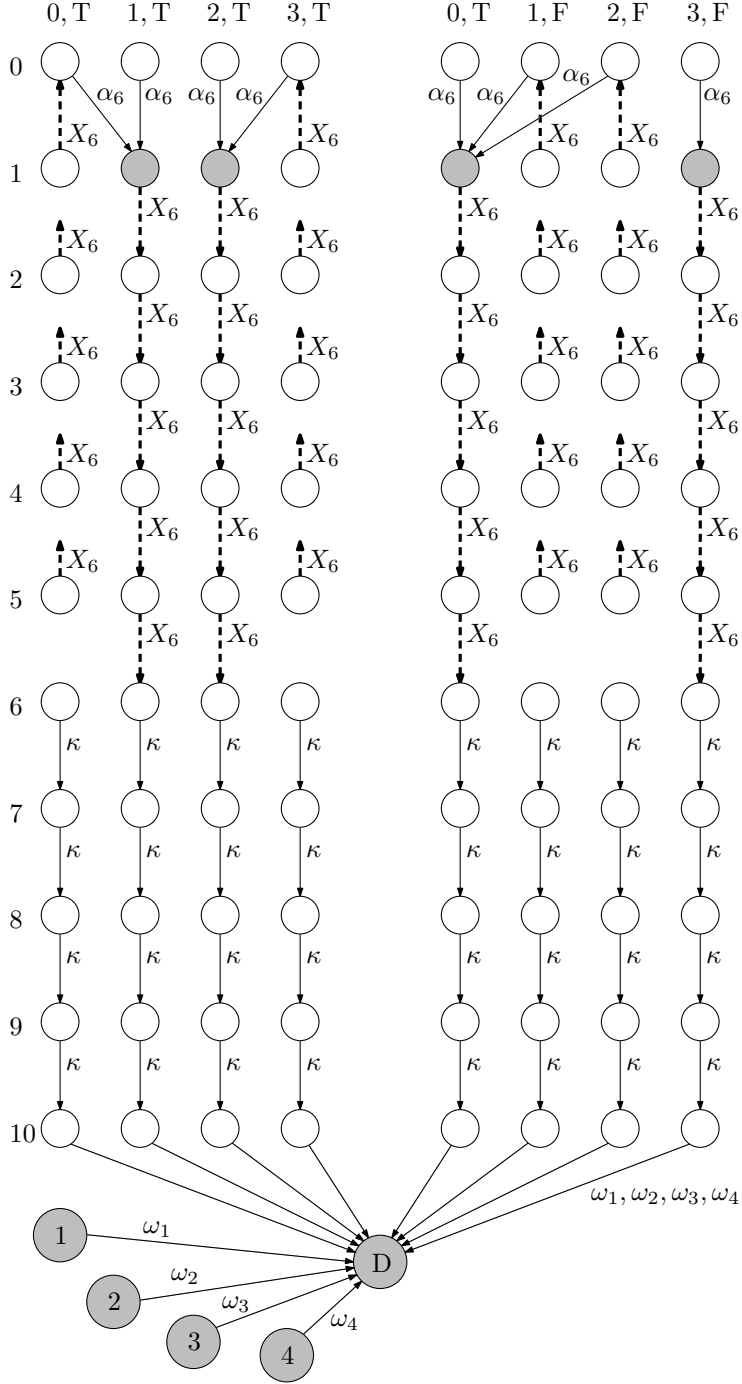


Fig. 1. Some transitions of the example automaton described on Page 9. The grey filling marks states that remain active after applying α_6 .

The actions of the letters $\alpha_1, \dots, \alpha_t$ should meet the following two ideas:

- Any reset word w of length $z(n) + 1$ have to start by some α_i .
- In such short reset word, right after the starting α_i , there must occur at least $z(n) - 1$ consecutive letters from X_i . Informally, by applying α_i we *choose* the automaton A_i .

How to do that? The number t may be quite large and each of $\alpha_1, \dots, \alpha_t$ needs to have a unique effect. The key tool is what we call *activity patterns*. Let us work with the set

$$R = \{0, \dots, q(t)\},$$

which matches „half of a row” of the guard table. Subsets of R correspond in a canonical way to binary representations of numbers $0, \dots, 2^{q(t)+1} - 1$. We will actually represent only the numbers $1, \dots, t$. These does not include any of the extreme values corresponding to the empty set and whole R , because we have $t < 2^{q(t)+1} - 1$. So let the mapping

$$\mathbf{b} : \{1, \dots, t\} \rightarrow 2^R$$

assign the corresponding subset of R to a number. For instance, it holds that

$$\mathbf{b}(11) = \{0, 1, 3\}$$

because $2^0 + 2^1 + 2^3$. For each $i = 1, \dots, t$ we define specific *pattern functions*

$$\pi_i^T, \pi_i^F : R \rightarrow R$$

such that

$$\begin{aligned} \text{rng } \pi_i^T &= \mathbf{b}(i), \\ \text{rng } \pi_i^F &= R \setminus \mathbf{b}(i) \end{aligned}$$

for each i . It is irrelevant how exactly are π_i^T and π_i^F defined. It is sure that they exist, because the range is never expected to be empty. Now the action of the letters $\alpha_1, \dots, \alpha_t$ is quite simple:

$$\begin{aligned} (h, k, T) &\xrightarrow{\alpha_i} (1, \pi_i^T(k), T) \\ (h, k, F) &\xrightarrow{\alpha_i} (1, \pi_i^F(k), F). \end{aligned}$$

for each $s \in \{1, \dots, n\}$ and each reasonable h, k .

Note that each α_i maps the entire guard table, and in particular the entire row 0, into the row 1. In fact, all „downward” transitions within the guard table will lead only one row down. And the only transitions escaping from the guard table will lead from the bottom row. Thus any reset word will have length at least $d' = z(n) + 1$. Moreover, during its application, in a time l the rows $0, \dots, l - 1$ will have to be all inactive. This is a key mechanism that the guard table uses for enforcing necessary properties of short reset words.

Let us define how the letters $x_{i,j}$ act on the guard table. Choose any $i \in \{1, \dots, t\}$. The action of $x_{i,j}$ within the guard table does not depend on j , all the letters coming from a single automaton act identically here:

– for the rows $h \in \{1, \dots, d_i\}$ we set

$$\begin{aligned} (h, k, T) &\xrightarrow{x_{i,j}} \begin{cases} (h+1, k, T) & \text{if } k \in \mathfrak{b}(i) \\ (0, k, T) & \text{otherwise} \end{cases} \\ (h, k, F) &\xrightarrow{x_{i,j}} \begin{cases} (h+1, k, F) & \text{if } k \notin \mathfrak{b}(i) \\ (0, k, T) & \text{otherwise} \end{cases} \end{aligned}$$

– and for the rows $h \in \{0\} \cup \{d_i + 1, \dots, z(n)\}$ we set

$$\begin{aligned} (h, k, T) &\xrightarrow{x_{i,j}} (0, k, T) \\ (h, k, F) &\xrightarrow{x_{i,j}} (0, k, F) \end{aligned}$$

Recall that sending an activity marker along any transition ending in the row 0 is a „suicide”. A word that does this cannot be a short reset word. So, if we restrict ourselves to letters from some X_i , the transitions defined above imply that that only in times $1, \dots, d_i$ the forthcoming letter can be some $x_{i,j}$. In the following $z(n) - d_i - 1$ steps the only letter from X_i that can be applied is κ .

The letter κ maps all the states of the guard table simply one state down, except for the rows 0 and $z(n)$. Set

$$\begin{aligned} (h, k, T) &\xrightarrow{\kappa} (h+1, k, T) \\ (h, k, F) &\xrightarrow{\kappa} (h+1, k, F) \end{aligned}$$

for each $h \in \{1, \dots, z(n) - 1\}$, and

$$\begin{aligned} (0, k, T) &\xrightarrow{\kappa} (0, k, T) \\ (0, k, F) &\xrightarrow{\kappa} (0, k, F) \\ (z(n), k, T) &\xrightarrow{\kappa} (0, k, T) \\ (z(n), k, F) &\xrightarrow{\kappa} (0, k, F). \end{aligned}$$

It remains to describe actions of the letters $\omega_1, \dots, \omega_n$ on the guard table. Just set

$$\begin{aligned} (z(n), k, T) &\xrightarrow{\omega} D \\ (z(n), k, F) &\xrightarrow{\omega} D \end{aligned}$$

for each k , and

$$\begin{aligned} (h, k, T) &\xrightarrow{\omega} (0, k, T) \\ (h, k, F) &\xrightarrow{\omega} (0, k, F) \end{aligned}$$

for each k in the remaining rows $h \in \{0, \dots, z(n) - 1\}$. Now the automaton A' is complete.

2.3 An Example

For instance, consider an input consisting of $t = 12$ automata A_1, \dots, A_{12} , each of them having $n = 4$ states. Because $z(4) = 10$ and $q(12) = 3$, the output automaton A' has 93 states in total. In Figure 1 all the states are depicted, together with some of the transitions. We focus on the transitions corresponding to the automaton A_6 , assuming that $d_6 = 5$.

The action of α_6 is determined by the fact that $6 = 2^1 + 2^2$ and thus

$$\begin{aligned} \text{rng } \pi_6^T &= \mathbf{b}(6) = \{1, 2\} \\ \text{rng } \pi_6^F &= R \setminus \mathbf{b}(6) = \{0, 3\}. \end{aligned}$$

If the first letter of a reset word is α_6 , after its application only the states

$$(1, 1, T), (1, 2, T), (1, 0, F), (1, 3, F)$$

remain active within the guard table. Now we need to move their activity markers one row down in each of the following $z(n) - 1 = 9$ steps. The only way to do this is to apply $d_6 = 5$ letters of X_6 and then $z(n) - 1 - d_6 = 4$ occurrences of κ . Then we are allowed to apply one of the letters $\omega_1, \dots, \omega_n$. But before that time, there should remain only one active state $s \in \{1, \dots, n\}$, so that we could use ω_s . The letter κ does not affect the activity within $\{1, \dots, n\}$ so we need to synchronize these states using $d_6 = 5$ letters from X_6 .

So, any short reset word of A' starting by α_6 has to contain a short reset word of A_6 .

2.4 The Guard Table Works

It remains to use the informally outlined ideas to prove that A' has the properties C1, C2, and C3 from Page 5.

Proof (C1). As it has been said by another words, for each letter $x \in X'$ and each state (h, k, Q) , where $Q \in \{T, F\}$ and $h \in \{0, \dots, z(n) - 1\}$, it holds that

$$(h, k, Q) \xrightarrow{x} (h', k', Q)$$

where $h' < h$ or $h' = h + 1$. So the shortest paths from the row 0 to the state D have length at least $z(n) + 1$.

Proof (C2). We should prove that any reset word w , having length exactly $z(n) + 1$, is of the form

$$w = \alpha_i y_1 \dots y_{d_i} \kappa^{z(n)-1-d_i} \omega_s,$$

such that, moreover, $y_1 \dots y_{d_i}$ is a reset word of A_i . The starting α_i is necessary, because $\alpha_1, \dots, \alpha_n$ are the only letters that map states from the row 0 to other rows. Denote the remaining $z(n)$ letters of w by $y_1, \dots, y_{z(n)}$.

Once an α_i is applied, there remain only $|R| = q(t) + 1$ active states in the guard table, all in the row 1, depending on i . Namely, the active states are exactly from

$$\{1\} \times \mathbf{b}(i) \times \{T\} \text{ and } \{1\} \times R \setminus \mathbf{b}(i) \times \{F\}$$

because this is exactly the range of α_i within the guard table. Let us continue by an induction. We claim that for $0 \leq \tau < d_i$ it holds what we have already proved for $\tau = 0$:

1. If $\tau \geq 1$, the letter y_τ lies in X_i . Moreover, if $\tau > d_i$, it holds that $w_\tau = \kappa$.
2. After the application of y_τ the active states within the guard table are exactly from

$$\{\tau + 1\} \times \mathbf{b}(i) \times \{\mathbf{T}\} \text{ and } \{\tau + 1\} \times R \setminus \mathbf{b}(i) \times \{\mathbf{F}\}.$$

For $i = 0$ both the claims hold. Take some $1 \leq \tau < d_i$ and suppose that the claims hold for $\tau - 1$. Let us use the second claim for $\tau - 1$ to prove the first claim for τ . So all the states from

$$\{\tau\} \times \mathbf{b}(i) \times \{\mathbf{T}\} \text{ and } \{\tau\} \times R \setminus \mathbf{b}(i) \times \{\mathbf{F}\}$$

are active. Which of the letters could appear as y_τ ? The letters $\omega_1, \dots, \omega_n$ and $\alpha_1, \dots, \alpha_t$ would map all the active states to the rows 0 and 1, which is a contradiction. Consider any letter $x_{k,j}$ for $k \neq i$. It holds that $\mathbf{b}(i) \neq \mathbf{b}(k)$, so there is some $c \in R$ lying in their symmetrical difference. For such c it holds that

$$(\tau, c, \mathbf{T}) \xrightarrow{x_{k,j}} (0, c, \mathbf{T}) \text{ if } c \in \mathbf{b}(i) \setminus \mathbf{b}(k)$$

or

$$(\tau, c, \mathbf{F}) \xrightarrow{x_{k,j}} (0, c, \mathbf{F}) \text{ if } c \in \mathbf{b}(k) \setminus \mathbf{b}(i)$$

which necessarily activates some state in the row 0, which is a contradiction again. So, $y_\tau \in X_i$. Moreover, if $\tau > d_i$, the letters from $X_i \setminus \{\kappa\}$ map the entire row τ into the row 0, so the only possibility is $y_\tau = \kappa$.

The letter y_τ maps all the active states right down to the row $\tau + 1$, so the second claim for τ holds as well.

Proof (C3). It is easy to verify that no „suicidal” transitions within the guard table are used, so during the application of

$$y_1 \dots y_{d_i} \kappa^{z(n)-1-d_i}$$

the activity markers just flow down from the row 1 to the row $z(n)$. Since $y_1 \dots y_{d_i}$ is a reset word of A_i , there also remains only one particular state s within $\{1, \dots, n\}$. Finally the letter ω_s is applied which maps s and the entire row $z(n)$ directly to D.

References

1. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. *Journal of Computer and System Sciences* 75(8), 423 – 434 (2009)
2. Černý, J.: Poznámka k homogénnym experimentom s konečnými automatmi. *Matematicko-fyzikálny časopis* 14(3), 208–216 (1964)

3. Eppstein, D.: Reset sequences for monotonic automata. *SIAM J. Comput.* 19(3), 500–510 (1990)
4. Fernau, H., Heggernes, P., Villanger, Y.: A multivariate analysis of some dfa problems. In: Dediu, A.H., Martín-Vide, C., Truthe, B. (eds.) *Language and Automata Theory and Applications, Lecture Notes in Computer Science*, vol. 7810, pp. 275–286. Springer Berlin Heidelberg (2013)
5. Grech, M., Kisielewicz, A.: The Černý conjecture for automata respecting intervals of a directed graph. *Discrete Mathematics & Theoretical Computer Science* 15(3), 61–72 (2013)
6. Pin, J.E.: On two combinatorial problems arising from automata theory. *Annals of Discrete Mathematics* 17, 535–548 (1983)
7. Sandberg, S.: Homing and synchronizing sequences. In: Broy, M., Jonsson, B., Katoen, J.P., Leucker, M., Pretschner, A. (eds.) *Model-Based Testing of Reactive Systems, Lecture Notes in Computer Science*, vol. 3472, pp. 5–33. Springer Berlin Heidelberg (2005)
8. Steinberg, B.: The Černý conjecture for one-cluster automata with prime length cycle. *Theoret. Comput. Sci.* 412(39), 5487 – 5491 (2011)
9. Trahtman, A.N.: Modifying the upper bound on the length of minimal synchronizing word. In: *FCT*. pp. 173–180 (2011)